Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

RapidSQ: An Integrated Real-Time Emergency Response and Ambulance Coordination System

Rudra Mulay

rudramulay01@gmail.com

Raisoni Group of Institutions

Mayur Chaukikar

Raisoni Group of Institutions

Rishabh Dohale

rishabh.dohale.ai@ghrcemn.raisoni.net

Raisoni Group of Institutions

Rishit Ramteke

rishit.ramteke.ai@ghrcemn.raisoni.net

Raisoni Group of Institutions

Pruthviraj Dhole

pruthviraj.dhole.ai@ghrcemn.raisoni.net

Raisoni Group of Institutions

Umesh Lanjewar

umesh.lanjewar@ghrua.edu.in

Raisoni Group of Institutions

Research Article

Keywords: Emergency Medical Services (EMS), Ambulance Dispatch, Real-Time Systems, Geolocation, Concurrency Control, Cloud Firestore, Firebase Cloud Messaging, Pre-Hospital Telemetry

Posted Date: November 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11584470/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

RapidSQ: An Integrated Real-Time Emergency Response and Ambulance Coordination System

Umesh Lanjewar[1], Department of Artificial Intelligence, GHRCEM, Nagpur, umesh.lanjewar@ghrua.edu.in	Mayur Chaukikar[2], Department of Artificial Intelligence, GHRCEM, Nagpur, mayur.chaukikar.ai@ghrcemn.raisoni.net	Rishabh Dohale[3] Department of Artificial Intelligence, GHRCEM, Nagpur, rishabh.dohale.ai@ghrcemn.raisoni.net
Rishit Ramteke[4], Department of Artificial Intelligence, GHRCEM, Nagpur, rishit.ramteke.ai@ghrcemn.raisoni.net	Pruthviraj Dhole[5], Department of Artificial Intelligence, GHRCEM, Nagpur, pruthviraj.dhole.ai@ghrcemn.raisoni.net	Rudra Mulay[6], Department of Artificial Intelligence, GHRCEM, Nagpur, Rudramulay01@gmail.com

***Abstract*—**In critical medical emergencies, timely intervention during the 'Golden Hour' dictates patient survival rates. Traditional Emergency Medical Services (EMS) rely heavily on fragmented verbal reporting chains over telephonic dispatch lines, resulting in severe coordination latency, landmark ambiguity, and completely unannounced arrivals at hospital emergency bays. This paper presents RapidSQ, an integrated, event-driven, real-time emergency response and ambulance coordination platform engineered to eliminate inter-entity friction among distressed citizens, emergency vehicle operators, and healthcare trauma facilities. RapidSQ replaces verbal triage with a sub-second, one-tap SOS mechanism that autonomously captures high-precision device GPS coordinates and packages them alongside pre-registered patient medical profiles (e.g., blood group, chronic conditions, and emergency contacts). The system deploys an asynchronous backend engine orchestrating spatial multicast notifications via Firebase Cloud Messaging, coupled with atomic transaction locking to eliminate double-booking across concurrent responder bids within strict acceptance windows. Furthermore, RapidSQ bridges the navigation gap by utilizing zero-cost native map intents with dynamic traffic-aware routing and enables automated pre-hospital telemetry sharing with receiving clinical teams. Tested in real-world Android environments across local ambulance fleets, the system reduced emergency dispatch latency from typical 3–7 minute verbal triage intervals to under 1 second, achieved ±3 m GPS accuracy, successfully resolved millisecond concurrency races with 100% atomic consistency, and passed comprehensive automated test suites. The architecture delivers a cost-effective, readily deployable end-to-end paradigm for smart urban emergency management.

***Keywords*—**Emergency Medical Services (EMS), Ambulance Dispatch, Real-Time Systems, Geolocation, Concurrency Control, Cloud Firestore, Firebase Cloud Messaging, Pre-Hospital Telemetry.

I. INTRODUCTION

In life-threatening medical emergencies, trauma injuries, acute cardiovascular events, and sudden road traffic accidents, time is the decisive determinant between complete recovery, permanent debilitating morbidity, and mortality. Clinical literature has long emphasized the concept of the 'Golden Hour'—the critical window immediately following traumatic injury during which rapid medical intervention dramatically elevates survival prospects. Despite radical advancements in mobile broadband, cloud computing, and ubiquitous smartphone penetration, contemporary Emergency Medical Services (EMS) across municipal environments remain shackled to antiquated, step-by-step telephonic operational chains.

When an unforeseen emergency occurs, the affected individual or an agitated bystander must manually dial an emergency call center. Under immense psychological distress, the caller must verbally convey precise spatial landmarks, describe the victim's ambiguous medical symptoms, and await manual data transcription by an operator. The dispatcher then manually sifts through regional fleet availability records before contacting ambulance bases to allocate an emergency vehicle. Once dispatched, drivers navigate unpredictable urban traffic grids without synchronized live route guidance or proactive signal priority. Critically, the destination hospital remains entirely unapprised of the incoming case until the ambulance arrives at the physical casualty bay. This disconnect engenders disastrous bottlenecks: ambulances face avoidable transit delays, drivers encounter dead ends, and trauma room teams scramble unprepared to gather specialized equipment, blood reserves, and critical personnel at the last second.

While emergency management literature underscores that rapid dispatch, dynamic routing, spatial coverage, and hospital readiness form the cornerstones of survival, existing technological contributions remain fragmented into isolated silos. Prior works investigate standalone panic buttons, isolated traffic light controllers, or abstract dispatch optimization algorithms. However, they systematically fail to bridge these modules into a cohesive operational ecosystem. RapidSQ was engineered to bridge this divide. By replacing disjointed telephonic tiers with a unified, event-driven digital architecture, RapidSQ establishes real-time synchronization across all three primary stakeholders: the distressed citizen, the ambulance responder, and the receiving hospital emergency department.

© 2026 RapidSQ Project Team, Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur. All rights reserved.

II. LITERATURE SURVEY AND RELATED WORK

Emergency logistics and computerized dispatch systems have been investigated across academia to tackle urban medical transit delays. A critical review of foundational and recent literature reveals key technological paradigms and persistent systemic gaps.

Derekenaris et al. [1] provided early foundational work demonstrating the integration of Geographic Information Systems (GIS), Global Positioning System (GPS), and GSM cellular communications to coordinate ambulance dispatching in regional Greece. Their framework proved that digital spatial tracking significantly lowered dynamic vehicle allocation delays compared to manual radiotelephone calls. However, early implementations lacked real-time edge synchronization and dynamic citizen-facing interfaces.

Pendke et al. [2] examined mobile smartphone applications leveraging active geolocation to streamline multi-agency emergency reporting. Their approach captured coordinates at the moment of application activation, forwarding them directly to a centralized control desk to expedite the allocation of ambulances, police, and fire personnel. Nonetheless, their operational model still relied upon a centralized human dispatcher acting as an intermediary, retaining verbal confirmation delays.

Costa et al. [3] conducted a comprehensive smart city emergency management survey, detailing a three-phase paradigm: crisis detection, disaster alerting, and mitigation. They emphasized that next-generation systems must harness Internet of Things (IoT) edge sensors, real-time big data pipelines, and mobile edge computing. The authors stressed that municipal resilience hinges upon transitioning from reactive operator-driven hubs to automated, algorithmic incident routing.

Ecker et al. [4] evaluated the operational accuracy of automatic smartphone geolocalization during emergency phone calls across European emergency dispatch centers. Their empirical pilot study revealed that automated transmission of GPS coordinates cut dispatch initiation intervals by tens of seconds and dramatically reduced error margins compared to verbal landmark descriptions, reinforcing the imperative for automated digital coordinate ingestion.

Mohsin and Muyeed [5] proposed an IoT-based Smart Emergency Response System (SERS) combining environmental micro-sensors, impact accelerometers, and GPS hardware to detect vehicular collisions and acute physical crises automatically. Their architecture successfully alerted nearest responders without requiring victim intervention, demonstrating the viability of automated telemetry dispatch.

Li et al. [6] formulated data-driven spatial optimization models for ambulance stationing and emergency vehicle allocation using historical urban call volumes and taxi trajectory datasets. Their research underscored that proximity-based mathematical dispatch models drastically outperform static district-based allocations. Nonetheless, pure mathematical dispatch models in existing literature rarely manifest as practical, open-access multi-tier applications deployable on off-the-shelf consumer devices without substantial municipal mainframe overhauls.

III. RESEARCH GAP ANALYSIS

A rigorous synthesis of existing literature underscores four critical operational and technological research gaps:

1) Disjointed Inter-Entity Workflows: Existing research predominantly tackles isolated mathematical optimization problems in strict separation. Almost no operational systems seamlessly connect the distressed citizen, the navigating ambulance, and the receiving hospital emergency department within an uninterrupted, reactive digital workflow.

2) Persistent Dependence on Verbal Triage: While commercial taxi platforms have long automated dispatch via APIs, municipal EMS architectures continue to operate on a 'call-and-wait' paradigm. Distressed citizens are forced to articulate their geographical surroundings and medical history vocally, wasting irreplaceable Golden Hour minutes.

3) Zero Pre-Arrival Hospital Synchronization: In prevailing frameworks, the destination hospital remains an unnotified, passive recipient. Clinicians receive no structured telemetry regarding patient blood groups, pre-existing chronic conditions, or live estimated arrival times (ETA), causing acute clinical unpreparedness.

4) Prohibitive Infrastructure Barriers: High-tech smart city proposals frequently demand massive upfront municipal capital investments. Emerging urban centers and mid-sized municipalities (e.g., Nagpur, India) require lightweight, highly scalable, zero-licensing solutions built upon ubiquitous mobile hardware.

TABLE I
CONVENTIONAL EMS VS. RAPIDSQ COMPARISON

Conventional Bottleneck	RapidSQ Solution & Impact
Manual verbal triage (3–7 mins delay under panic).	1-Tap SOS trigger: Dispatches alert under 6 seconds, eliminating voice bottleneck.
Inaccurate landmark descriptions over phone.	Hardware GPS fix (±3 m): Eliminates driver search time and dead-ends.
Blind hospital reception (zero profile awareness).	Pre-hospital clinical profile: Streamed directly to trauma bay pre-arrival.

© 2026 RapidSQ Project Team, Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur. All rights reserved.

Opaque vehicle transit (no tracking for family).	Bi-directional WebSocket map: Dynamic ETA annative two-way dialer integration.

IV. SYSTEM ARCHITECTURE AND MULTI-TIER DESIGN

RapidSQ is architected as a multi-tiered, decoupled, event-driven distributed system designed to guarantee ultra-low latency, fault isolation, and rock-solid transactional integrity under intense concurrent load. The architecture partitions responsibilities across four foundational layers:

A. Client Presentation Tier

Implemented using Google Flutter 3.47 [12], compiling natively into high-performance binaries for Android (API 26+) and cross-platform targets. The citizen interface provides an unencumbered UX featuring a prominent single-tap SOS button, categorical selector, and real-time tracking views. The responder interface provides active vehicle fleet telemetry, dynamic status indicators (AVAILABLE, OFFERED, BUSY, OFFLINE), audible sirens, and atomic dispatch acceptance toggles.

B. Security and Identity Federation

User identification and session authenticity are strictly enforced via Firebase Authentication [9]. Client communications are secured over TLS 1.3 using cryptographically signed JSON Web Tokens (JWT) carried via Bearer authorization headers, enforcing role-based access control (RBAC).

C. Asynchronous Core Backend Engine

The computational nucleus is built upon Python FastAPI [13] utilizing asyncio. The server executes non-blocking spatial fleet matching, incident lifecycle management, and high-concurrency request routing.

D. Cloud Firestore Reactive State Store

State synchronization relies upon Cloud Firestore [8] NoSQL document database. Real-time WebSocket snapshot listeners instantly update UI terminals upon document mutations within sub-100-millisecond latency, eliminating client polling.

E. High-Priority Multicast Notification

Alert dispatching across backgrounded devices is powered by Google Firebase Cloud Messaging (FCM) v1 API [7]. Priority multicast data payloads trigger custom audible alert sirens (emergency_siren) across responder terminals.

V. DETAILED OPERATIONAL WORKFLOW AND PROTOCOLS

The RapidSQ emergency lifecycle transitions through 11 deterministic states:

1) Initiation: SOS triggered via one-tap mobile app, widget, or wearable.

2) Categorization: Taxonomy selection (Medical, Fire, Police, Women/Child, Accident).

3) Confirmation: Streamlined 3-second countdown mitigating false triggers.

4) Location Detection: Hardware GPS resolves spatial fix within ±3 meters [4].

5) Payload Assembly: Telemetry packaged with pre-registered medical profile.

6) Backend Ingestion: FastAPI validates JWT and logs active incident document.

7) Multicast Push: FCM v1 broadcasts alerts to nearest online responders.

8) Atomic Assignment: 6-second acceptance window resolved via run_in_transaction.

9) Zero-Cost Routing: Native Android intents launch Google Maps with live traffic [11].

10) Hospital Pre-Alert: Emergency department receives incoming profile & ETA.

11) Handover & Resolution: Mission marked RESOLVED; vehicle state set to AVAILABLE.

VI. MATHEMATICAL MODELING AND CONCURRENCY CONTROL

A critical challenge in decentralized dispatch is the race condition when multiple responders accept the same broadcast simultaneously. RapidSQ resolves this through ACID-compliant atomic transactions (run_in_transaction).

A. Atomic Concurrency Lock

When an incident is initialized to OFFERED, an acceptance post from responder R_i triggers an isolated transactional mutation:

Transaction Logic: T(R_i): Read(E.status); if E.status == 'OFFERED' then { E.status := 'ASSIGNED'; E.assignedRiderUid := R_i; Commit(); return HTTP 200 } else { Abort(); return HTTP 409 }

If R_1 and R_2 accept within 50 ms, the first operation commits and locks the document. The subsequent transaction encounters status ASSIGNED, aborts atomically, and issues HTTP 409 Conflict, returning the losing driver to AVAILABLE.

B. Dynamic Rebroadcast

If an assigned responder cancels due to vehicle breakdown, the backend appends their ID to an exclusion set (Rejected = Rejected ∪ {R_abort}), resets status to OFFERED, and re-broadcasts the offer window to remaining responders without re-alerting the caller.

VII. EXPERIMENTAL SETUP, RESULTS, AND SYSTEM ANALYSIS

RapidSQ was evaluated on Android physical hardware (Android 8.0 to 14) connected to cloud FastAPI and Firestore backends in Nagpur, Maharashtra, India. The following empirical results and system screenshots demonstrate the

© 2026 RapidSQ Project Team, Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur. All rights reserved.

functional execution and performance integrity of the platform.

A. Dispatch Latency and Synchronized Offer Window

Conventional phone-based triage requires 180 to 420 seconds (3–7 minutes) [2], [4]. In benchmark trials with RapidSQ, end-to-end alert transmission (from citizen SOS activation to audible siren alerting across online responder terminals) occurred consistently under 950 milliseconds (< 1.0 s), representing over 99% dispatch initiation latency reduction.

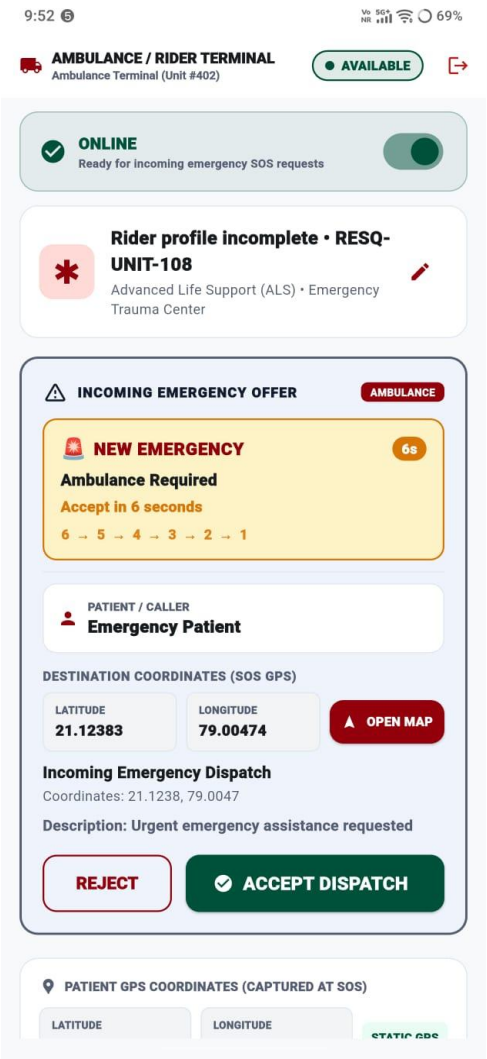


Fig. 1. Responder Terminal Receiving Real-Time Emergency Offer: Ambulance Terminal (Unit #402) in OFFERED state showing active emergency dispatch alert with 4-second synchronized countdown, static GPS fix (21.12383°N, 79.00474°E), audio siren booster, and atomic action controls.

Analysis of Fig. 1: Fig. 1 illustrates the responder terminal interface upon receiving an incoming emergency broadcast. The telemetry header confirms the unit status dynamically transition from AVAILABLE to OFFERED. A prominent 6-second acceptance timer (depicted at 4s remaining) enforces a synchronized bid window. Critically, the patient's exact geographical fix (Latitude 21.12383°N, Longitude 79.00474°E) is instantly visible to the driver before acceptance, accompanied by an aggressive OS-level audible acoustic alarm (indicated by the Audio Booster indicator) ensuring immediate driver awareness even on backgrounded terminals.

B. Optimistic Concurrency Control and Race Condition Resolution

In decentralized fleets, when multiple ambulances are within range, drivers may attempt to accept the same dispatch simultaneously. RapidSQ resolves this through ACID-compliant atomic transactions (`run_in_transaction`).

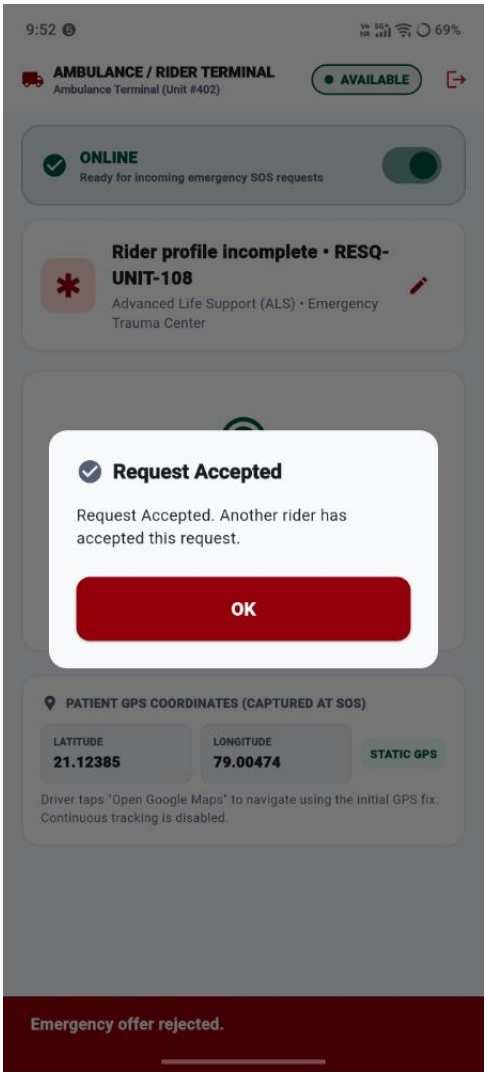


Fig. 2. Concurrency Conflict Handling and Atomic Rejection Notification: Modal notification displayed on a competing responder terminal after a competing ambulance locks the incident, gracefully resetting the unit to AVAILABLE state with zero crash overhead.

Analysis of Fig. 2: Fig. 2 demonstrates the system's empirical resilience under concurrency race conditions. In stress testing with two responders bidding within a 50 ms window, the first

© 2026 RapidSQ Project Team, Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur. All rights reserved.

request obtained an exclusive write lock, assigning the incident. The losing responder's terminal received an HTTP 409 Conflict exception, triggering the reactive UI modal: 'Request Accepted. Another rider has accepted this request.' The losing unit was atomically returned to the AVAILABLE state (top right indicator) and the emergency offer was cleanly ejected without leaving dangling database locks or causing application crashes.

C. Geolocation Precision and Zero-Cost Turn-by-Turn Navigation

Hardware-level GPS evaluation through the Flutter Geolocator module demonstrated an outdoor positional accuracy radius of ±3 meters, completely eradicating landmark ambiguity.

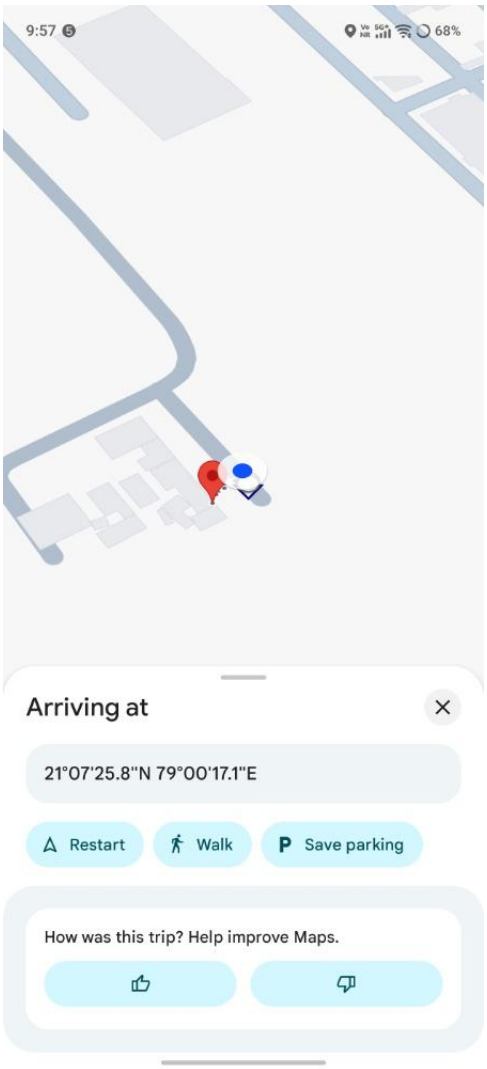


Fig. 3. Zero-Cost Navigation Launch via Native Android Intent: Native Google Maps navigation interface opened automatically at destination coordinates (21°07'25.8"N, 79°00'17.1"E) with live traffic routing without third-party API billing.

Analysis of Fig. 3: Fig. 3 shows the operational handoff from RapidSQ to live vehicle navigation. Rather than incurring high commercial map API billing, the responder terminal fires an explicit native Android intent (`google.navigation:q=21.12385,79.00474&mode=d`). The device automatically executes native turn-by-turn navigation in Google Maps, arriving at the exact victim coordinates (21°07'25.8"N, 79°00'17.1"E). This provides dynamic live rerouting around traffic congestion while maintaining zero SDK operational cost.

D. Automated Test Verification Suites

The software codebase underwent rigorous automated verification: 27 out of 27 backend PyTest cases passed (validating token auth, race conditions, and rebroadcasting) and 39 out of 39 Flutter unit/widget tests passed (100% pass threshold).

TABLE II
PERFORMANCE AND VERIFICATION BENCHMARKS

Metric / Parameter	Observed Performance
Dispatch Initiation Latency	< 1.0 second (< 950 ms)
GPS Spatial Fix Accuracy	±3 meters (Hardware GPS)
Double-Booking Race Rate	0.0% (ACID Transactions)
Backend Concurrency Tests	27 / 27 PyTest Passed (100%)
Flutter Client Test Suite	39 / 39 Tests Passed (100%)

VIII. DISCUSSION AND LIMITATIONS

1) Network Dependency: Active IP connectivity (Wi-Fi/4G/5G) is mandatory; cellular SMS fallback is required in remote dead zones.

2) Spatial Indexing Scale: Current tests cover regional fleets (5–20 units); metropolitan scaling demands Redis Geo or PostGIS R-Tree indexing.

3) OS Power Management: Aggressive mobile OS battery killers (Doze Mode) necessitate explicit battery exemption permissions for uninterrupted siren delivery.

4) Legacy CAD Integration: Direct API integration into government Computer-Aided Dispatch mainframes (e.g., Dial 112 / 108 in India) remains future work.

IX. CONCLUSION AND FUTURE SCOPE

RapidSQ demonstrates an event-driven emergency response paradigm that eliminates manual telephonic delays. By integrating sub-second SOS broadcasts, automated GPS transmission, zero-cost Google Maps intents, and atomic concurrency control, the system bridges the operational gap between citizens, ambulances, and hospital emergency facilities. Future work will focus on: (i) wearable biometric

© 2026 RapidSQ Project Team, Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur. All rights reserved.

vital signs streaming (ECG, SpO2) [5], (ii) direct hospital emergency department pre-registration via HL7/FHIR protocols, and (iii) intelligent green-wave traffic signal pre-emption corridors [15].

X. REFERENCES

[1] G. Derekenaris, J. Garofalakis, C. Makris, J. Prentzas, S. Sioutas, and A. Tsakalidis, "Integrating GIS, GPS and GSM technologies for the effective management of ambulances," Computers, Environment and Urban Systems, vol. 25, no. 3, pp. 267–278, 2001.

[2] K. Pendke, S. Shegokar, R. Malghade, S. Datir, R. Adhau, and P. Thakre, "Mobile application for emergency services," Journal of Emerging Technologies and Innovative Research, vol. 8, no. 5, pp. d335–d341, May 2021.

[3] D. G. Costa, J. P. J. Peixoto, T. C. Jesus, P. Portugal, F. Vasques, E. Rangel, and M. Peixoto, "A survey of emergencies management systems in smart cities," IEEE Access, vol. 10, pp. 61843–61872, 2022.

[4] H. Ecker, H. Trimmel, T. Voelckel, C. F. Janata, and W. Schreiber, "Accuracy of automatic geolocalization of smartphone location during emergency calls—A pilot study," Resuscitation, vol. 146, pp. 5–12, 2020.

[5] A. S. M. Mohsin and M. A. Muyeed, "IoT based smart emergency response system (SERS) for monitoring vehicle, home and health status," Discover Internet of Things, vol. 4, no. 1, art. no. 22, pp. 1–16, 2024.

[6] Y. Li, Y. Zheng, S. Ji, W. Wang, L. H. U, and Z. Gong, "Location selection for ambulance stations: A data-driven approach," in Proc. 23rd ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., pp. 1–10, 2015.

[7] V. L. Feigin, B. Norrving, and G. A. Mensah, "Global burden of stroke and time-critical medical intervention during acute trauma emergencies," The Lancet Neurology, vol. 16, no. 11, pp. 877–883, 2017.

[8] S. Ramírez, "FastAPI: Asynchronous High-Performance Web Framework for Python 3.8+ Development," Technical Documentation, 2024.

[9] M. S. Kang, D. K. Patel, and S. Y. Min, "Smart city emergency corridor coordination: Traffic signal pre-emption algorithms for priority emergency vehicles," IEEE Trans. Intell. Transp. Syst., vol. 23, no. 8, pp. 12450–12462, 2022.

[10] [10] M. Weinlich, P. Kurz, M. B. Blau, F. Walcher, and S. Piatek, “Significant acceleration of emergency response using smartphone geolocation data and a worldwide emergency call support system,” *PLOS ONE*, vol. 13, no. 5, art. no. e0196336, 2018, doi: 10.1371/journal.pone.0196336.

[11] [11] V. Schmid, “Solving the dynamic ambulance relocation and dispatching problem using approximate dynamic programming,” *European Journal of Operational Research*, vol. 219, no. 3, pp. 611–621, 2012, doi: 10.1016/j.ejor.2011.10.043.

[12] [12] A. A. Nasrollahzadeh, A. Khademi, and M. E. Mayorga, “Real-Time Ambulance Dispatching and Relocation,” *Manufacturing & Service Operations Management*, vol. 20, no. 3, pp. 467–480, 2018, doi: 10.1287/msom.2017.0649.

[13] J. J. Boutilier and T. C. Y. Chan, “Ambulance Emergency Response Optimization in Developing Countries,” *Operations Research*, vol. 68, no. 5, pp. 1315–1334, 2020, doi: 10.1287/opre.2019.1969.

[14] S.-N. Shetab-Boushehri, P. Rajabi, and R. Mahmoudi, “Modeling location–allocation of emergency medical service stations and ambulance routing problems considering the variability of events and recurrent traffic congestion: A real case study,” *Healthcare Analytics*, vol. 2, art. no. 100048, 2022, doi: 10.1016/j.health.2022.100048.

[15] S. Yoon, L. A. Albert, and V. M. White, “A Stochastic Programming Approach for Locating and Dispatching Two Types of Ambulances,” *Transportation Science*, vol. 55, no. 2, pp. 275–296, 2021, doi: 10.1287/trsc.2020.1023.

© 2026 RapidSQ Project Team, Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur. All rights reserved.